Python: function averager

cdutil.averager = averager(V, axis=None, weights=None, action='average', returned=0, weight=None, combinewts=None)

```
                              Documentation for averager():
                              -----------------------------
    The averager() function provides a convenient way of averaging your data g
    you control over the order of operations (i.e which dimensions are average
    over first) and also the weighting for the different axes. You can pass yo
    own array of weights for each dimension or use the default (grid) weights
    specify equal weighting.


    Author: Krishna AchutaRao : achutarao1@llnl.gov

    Returns:
    -------
           The average over the specified dimensions.
    Usage:
    ------
    from cdutil import averager
    averager( V, axis='axisoptions', weights=weightoptions, action='average',
              returned='0')

    Where V is an array. It can be an array of Numeric, MA or MV type. In each
    the function returns an array (except when it results in a scalar) of the
    type as V. See examples for more details.

    Optional Arguments:
    -------------------
    axis=axisoptions
            Restrictions: axisoptions has to be a string
            Default : first dimension in the data you pass to the function.

            You can pass axis='tyx', or '123', or 'x (plev)' etc.  the same wa
            in order= options for variable operations EXCEPT that
            '...'(i.e Ellipses) are not allowed. In the case that V is a Numer
            MA array, axis names have no meaning and only axis indexes are val


    weights=weightoptions
            Default :
                     'weighted' for Transient Variables (MVs)
                     'unweighted' for MA or Numeric arrays.

            Note that depending on the array being operated on by averager, th
            default weights change!

            Weight options are one of 'weighted', 'unweighted',  an array of w
            each dimension or a MaskedVariable of the same shape as the data x

            - 'weighted' means use the grid information to generate weights fo
```

```
          that dimension.

        - 'unweighted' means use equal weights for all the grid points in

        - Also an array of weights (of the same shape as the dimension bei
          averaged over or same shape as V) can be passed.

          Additional Notes on 'weighted' option: The weights are generated
          using the bounds for the specified axis. For latitude and Longit
          the weights are calculated using the area (see the cdms manual
          grid.getWeights() for more details) whereas for the other axes
          weights are the difference between the bounds (when the bounds a
          available). If the bounds are stored in the file being read in,
          those values are used. Otherwise, bounds are generated as long a
          cdms.setAutoBounds('on') is set. If cdms.setAutoBounds() is set
          'off', then an Error is raised.

action='average' or 'sum'
        Default : 'average'

        You can either return the weighted average or the weighted sum of
        data by specifying the keyword argument action=

returned = 0 or 1
        Default: 0

        - 0 implies sum of weights are not returned after averaging operat
        - 1 implies the sum of weights after the average operation is retu

combinewts = None, 0 or 1
        Default: None - same as 0
        - 0 implies weights passed for individual axes are not combined in
          weight array for the full variable V before performing operation
       - 1 implies weights passed for individual axes are combined into on
         weight array for the full variable before performing average or s
         operations. One-dimensional weight arrays or key words of 'weight
         'unweighted' must be passed for the axes over which the operation
         to be performed. Additionally, weights for axes that are not bein
         averaged or summed may also bepassed in the order in which they a
         If the weights for the other axes are not passed, they  are assum
         be equally weighted.

Examples:
---------
        >>> f = cdms.open('data_file_name')
        >>> averager(f('variable_name'), axis='1')
        # extracts the variable 'variable_name' from f and averages over t
        # dimension whose position is 1. Since no other options are specif
        # defaults kick in i.e weight='weighted' and returned=0

        >>> averager(V, axis='xy', weights=['weighted','unweighted'])
        or
```

```
>>> averager(V, axis='t', weights='unweighted')
or
>>> averager(V, axis='x')
# Default weights option of 'weighted' is implemented
or
>>> averager(V, axis='x', weights=mywts)
# where mywts is an array of shape (len(xaxis)) or shape(V)
or
>>> averager(V, axis='(lon)y', weights=[myxwts, myywts])
# where myxwts is of shape len(xaxis) and myywts is of shape len(y
or
>>> averager(V, axis='xy', weights=V_wts)
# where V_wts is a Masked Variable of shape V
or
>>> averager(V, axis='x', weights='unweighted', action='sum')
# will return the equally weighted sum over the x dimension
or
>>> ywt = area_weights(y)
>>> fractional_area = averager(ywt, axis='xy',
                           weights=['unweighted', 'unweighted'], acti
# is a good way to compute the area fraction that the
# data y that is non-missing
```

Note:

```
When averaging data with missing values, extra care needs to be ta
It is recommended that you use the default weights='weighted' opti
This uses cdutil.area_weights(V) to get the correct weights to
pass to the averager.
>>> averager(V, axis='xy', weights='weighted')

The above is equivalent to:
>>> V_wts = cdutil.area_weights(V)
>>> result = averager(V, axis='xy', weights=V_wts)
or
>>> result = averager(V, axis='xy', weights=cdutil.area_weights(V)

However, the area_weights function requires that the axis bounds a
stored or can be calculated (see documentation of area_weights for
details). In the case that such weights are not stored with the ax
specifications (or the user desires to specify weights from anothe
source), the use of combinewts option can produce the same results
In short, the following two are equivalent:
>>> xavg_1 = averager(X, axis = 'xy', weights = area_weights(X))
>>> xavg_2 = averager(X, axis = 'xy', weights = ['weighted', 'weig

Where X is a function of x, y and a third dimension such as time o

In general, the above can be substituted with arrays of weights wh
the 'weighted' keyword appears.
```